global Feature EXtraction

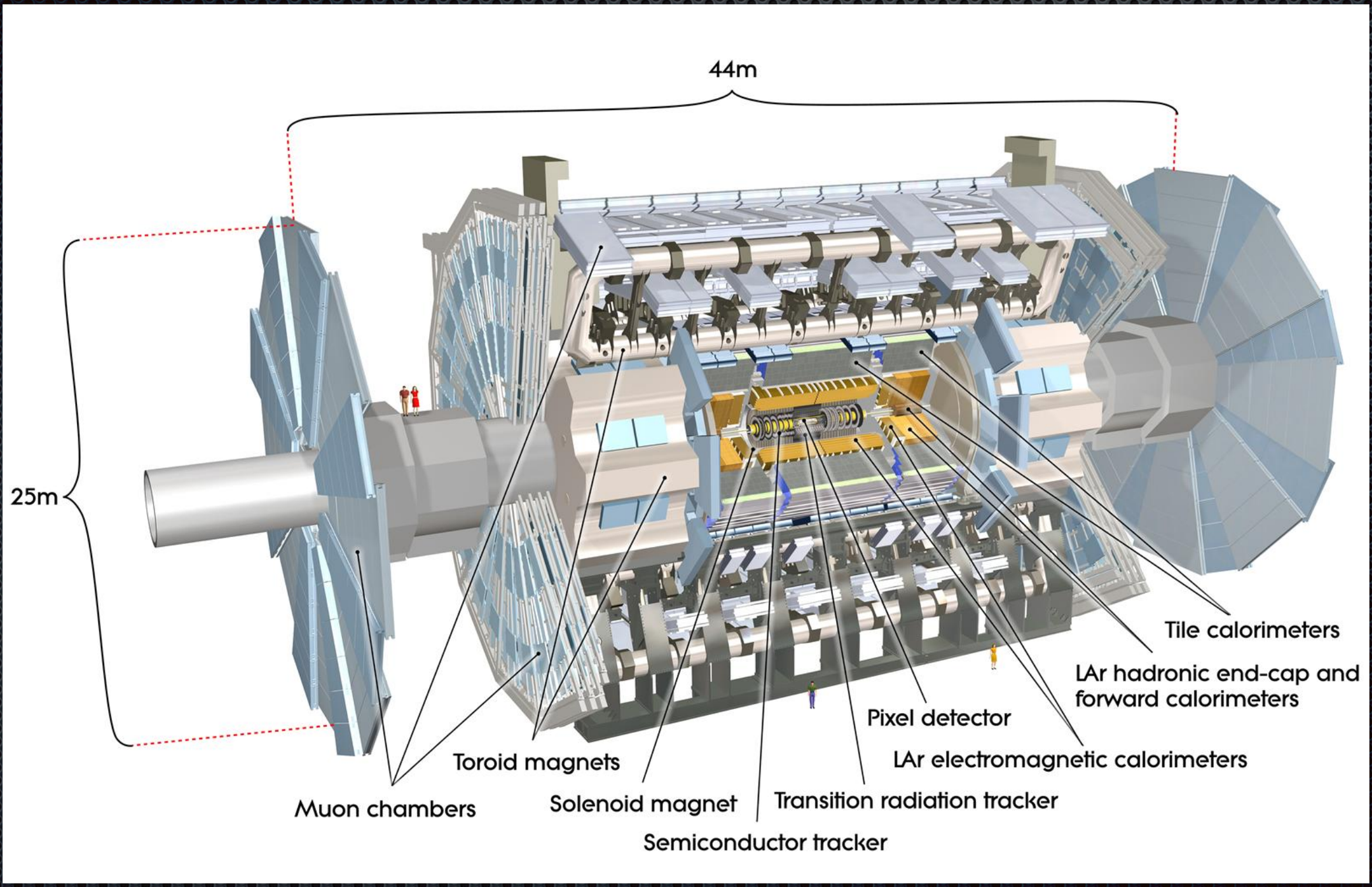
Developing Slow Control and Monitoring Software for a Phase-I Trigger Upgrade of the ATLAS Experiment

Giordon Stark
DPF 2017
giordonstark.com

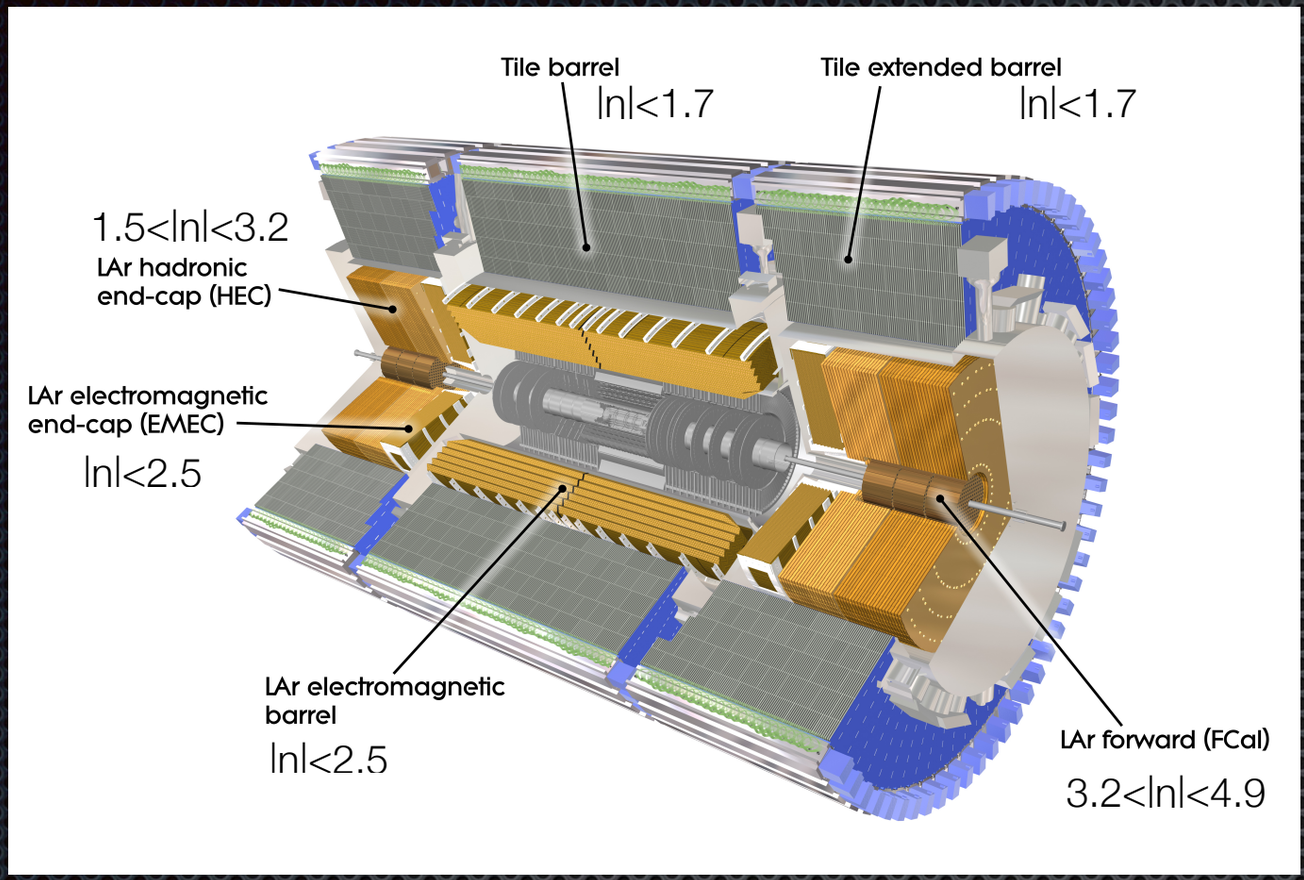
https://gfex.cern.ch

gFEX

The ATLAS Detector



Calorimetry and Trigger

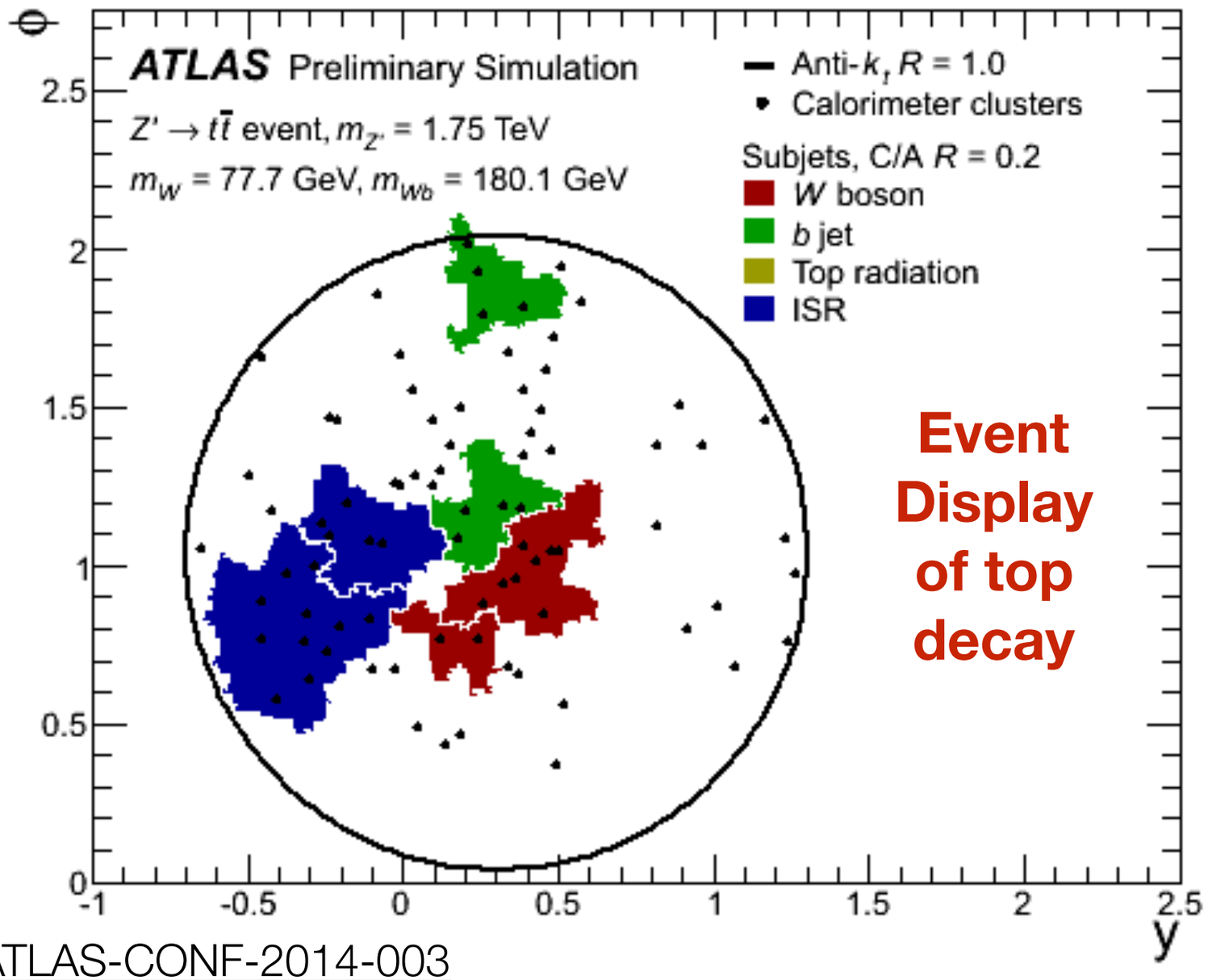


- 2 main components to locate and absorb particles: Liquid Argon and Tile Hadronic Calorimeters

- The trigger system uses data from the calorimeters
- Bunches of protons collide every 25 ns (**40 MHz** rate)
  - Need to reduce this rate to **~1 kHz** for writing to disk
- **Goal**: retain efficiency of processes sought for in ATLAS
  - Need a lot of smart rejection
  - Need it fast and performant

The Motivation (I)

- High p_T Lorentz-boosted top quarks, W/Z/h bosons, and exotics are critical elements of the ATLAS physics program



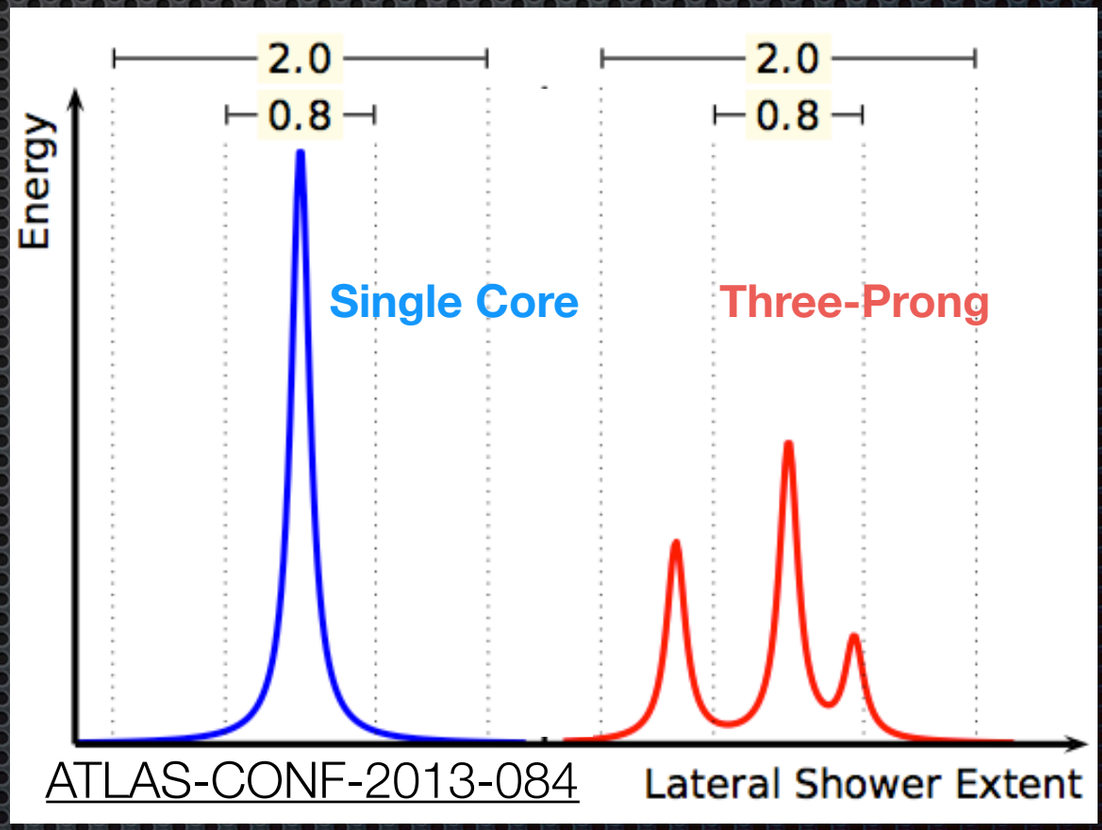
ATLAS-CONF-2014-003

- As luminosity increases, the trigger thresholds go up
- Signal/Noise will increase with increased luminosity

Can we be smarter?

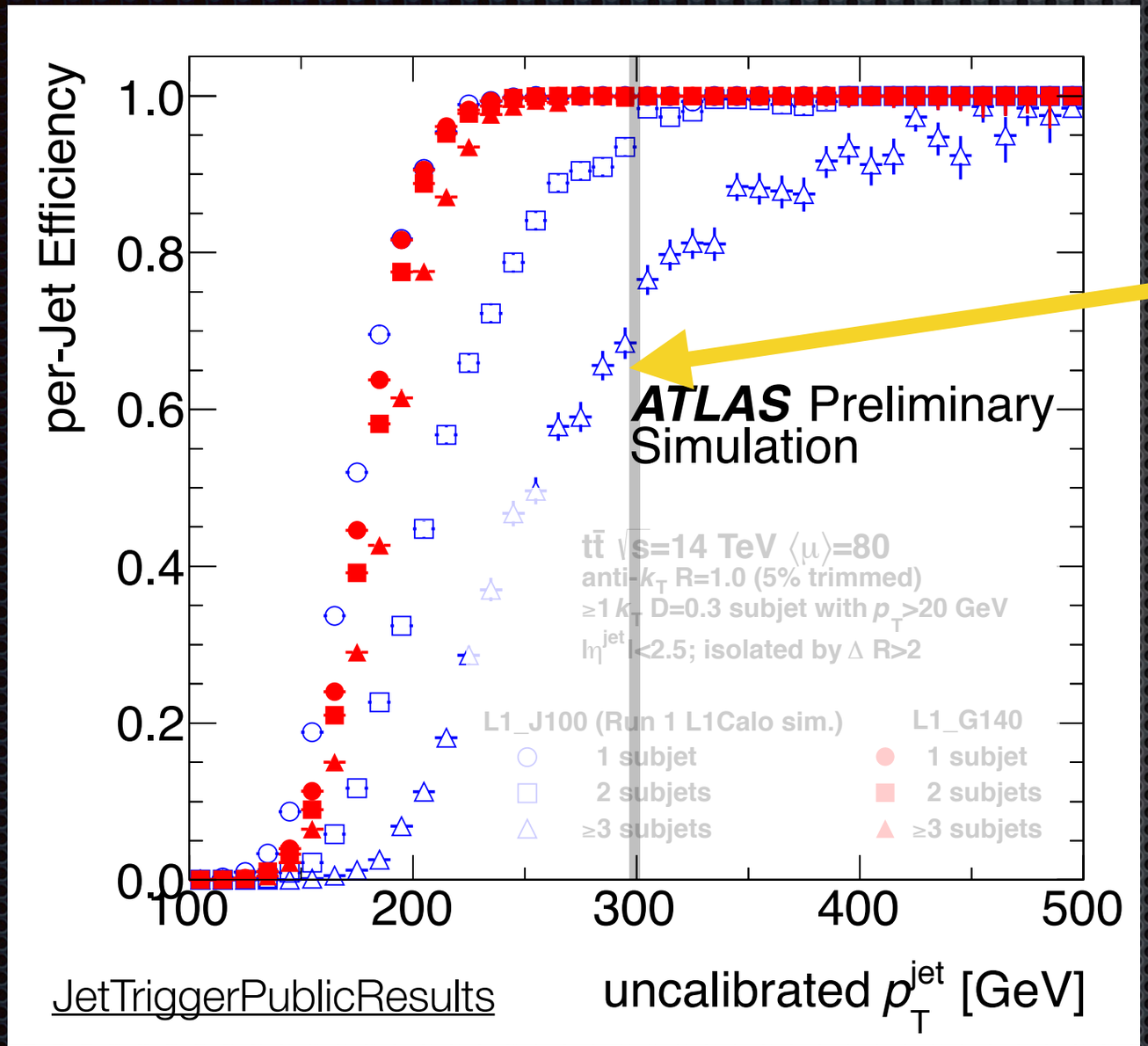
The Motivation (II)

- Current trigger uses a small window to quickly scan an event
  - If a jet with sufficient energy decays over too large of an area, trigger will not fire
- Can we increase the trigger region?
  - **Yes, with gFEX**



The Motivation (III)

Blue=Current Trigger @ 100 GeV



a top quark at 300 GeV

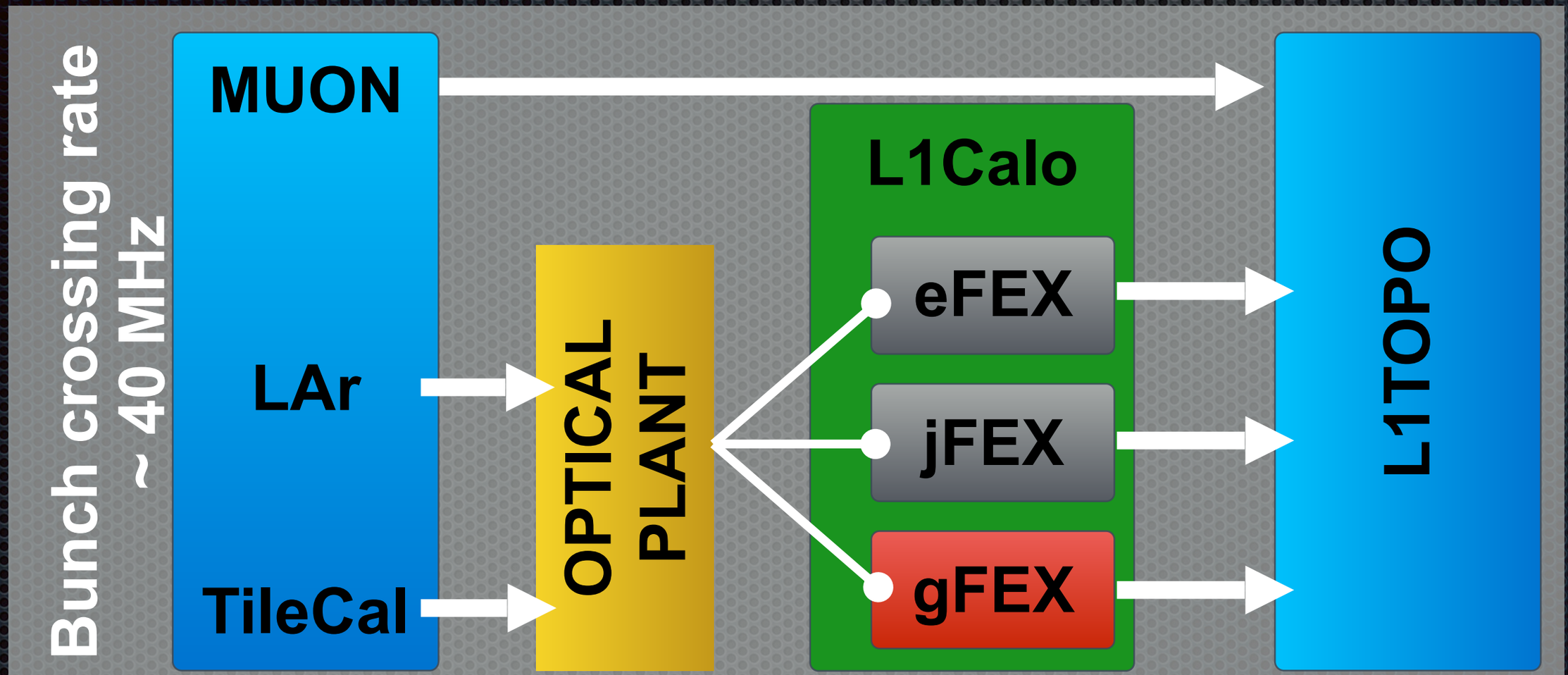
- Many analyses in ATLAS are sensitive to boosted objects with substructure
  - would like a trigger that does not cut them away
- gFEX maintains a flat trigger efficiency here

Red=gFEX Trigger @ 140 GeV

Current trigger is inefficient for jets with significant substructure

global Feature Extraction

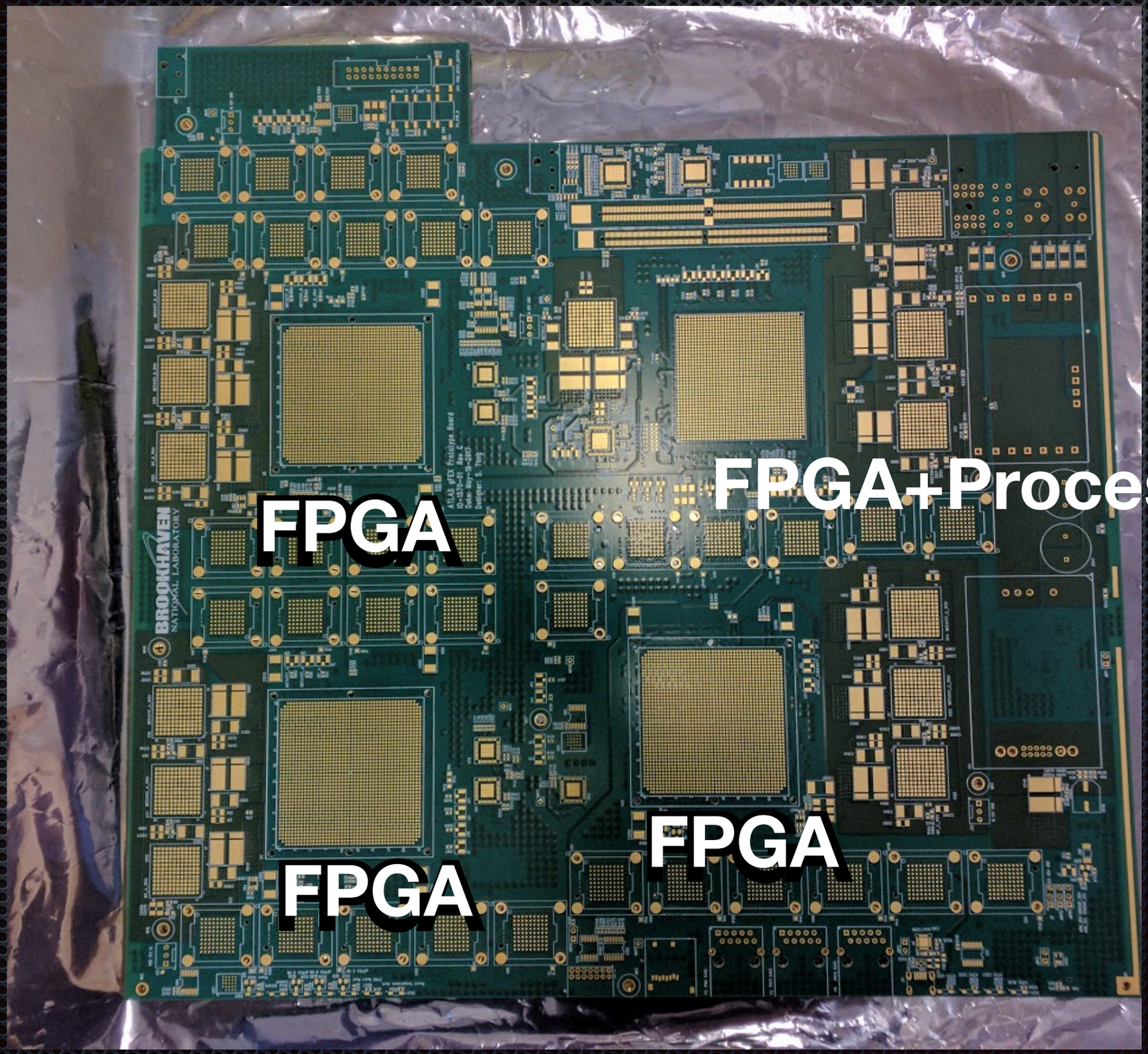
LHC Run 3 — a new feature extraction module



Our Solution: increase the RoI and processing speed, but some loss in angular resolution

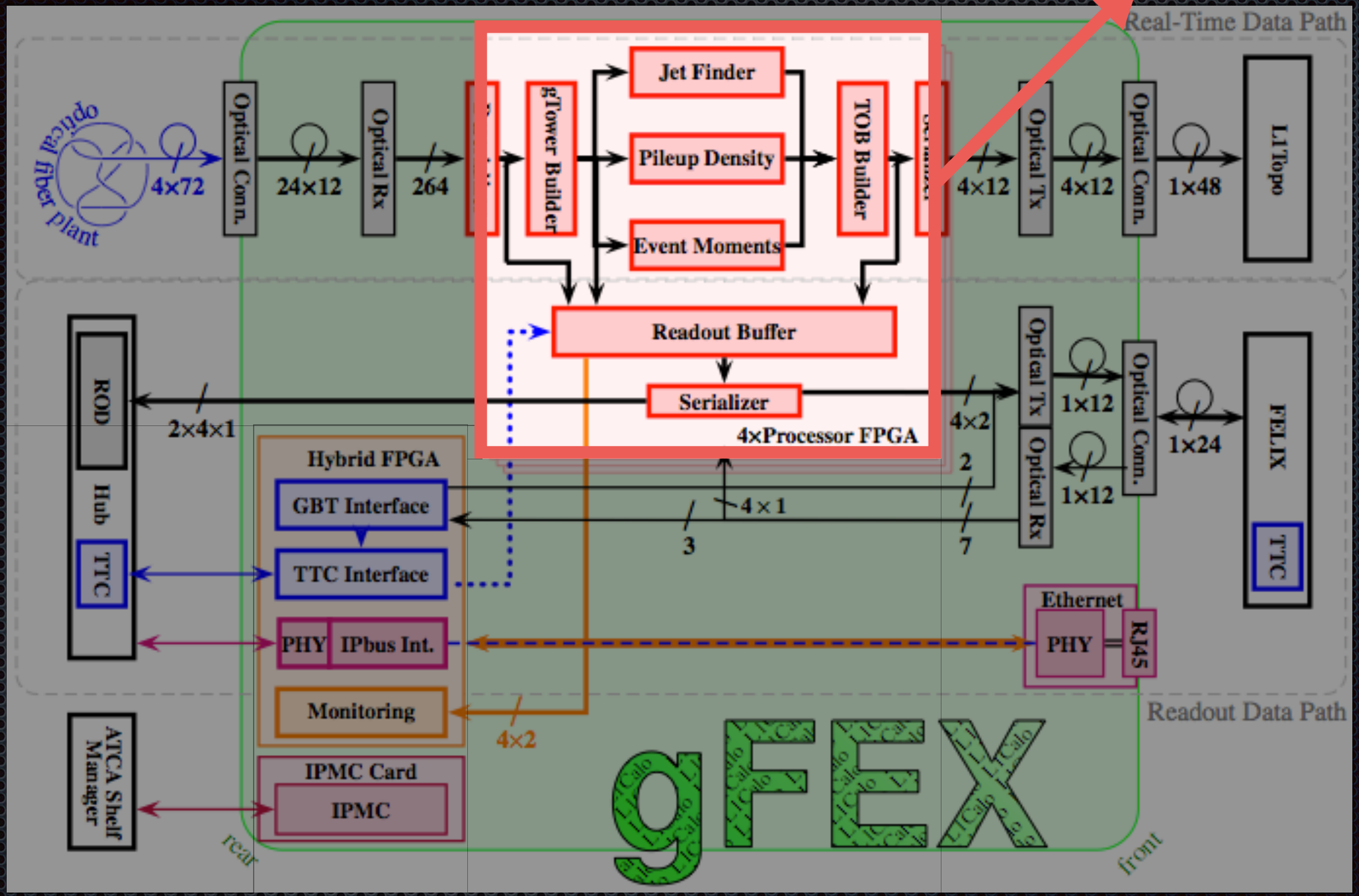
- **algorithms run within 5 bunch crossings** (125 ns), not including data input/output
- L1Topo/HLT get info about **jets above a threshold and pileup calculation** for other triggers
- **full calorimeter information on a single board** enables calculation of global event quantities

gFEX Prototype Board



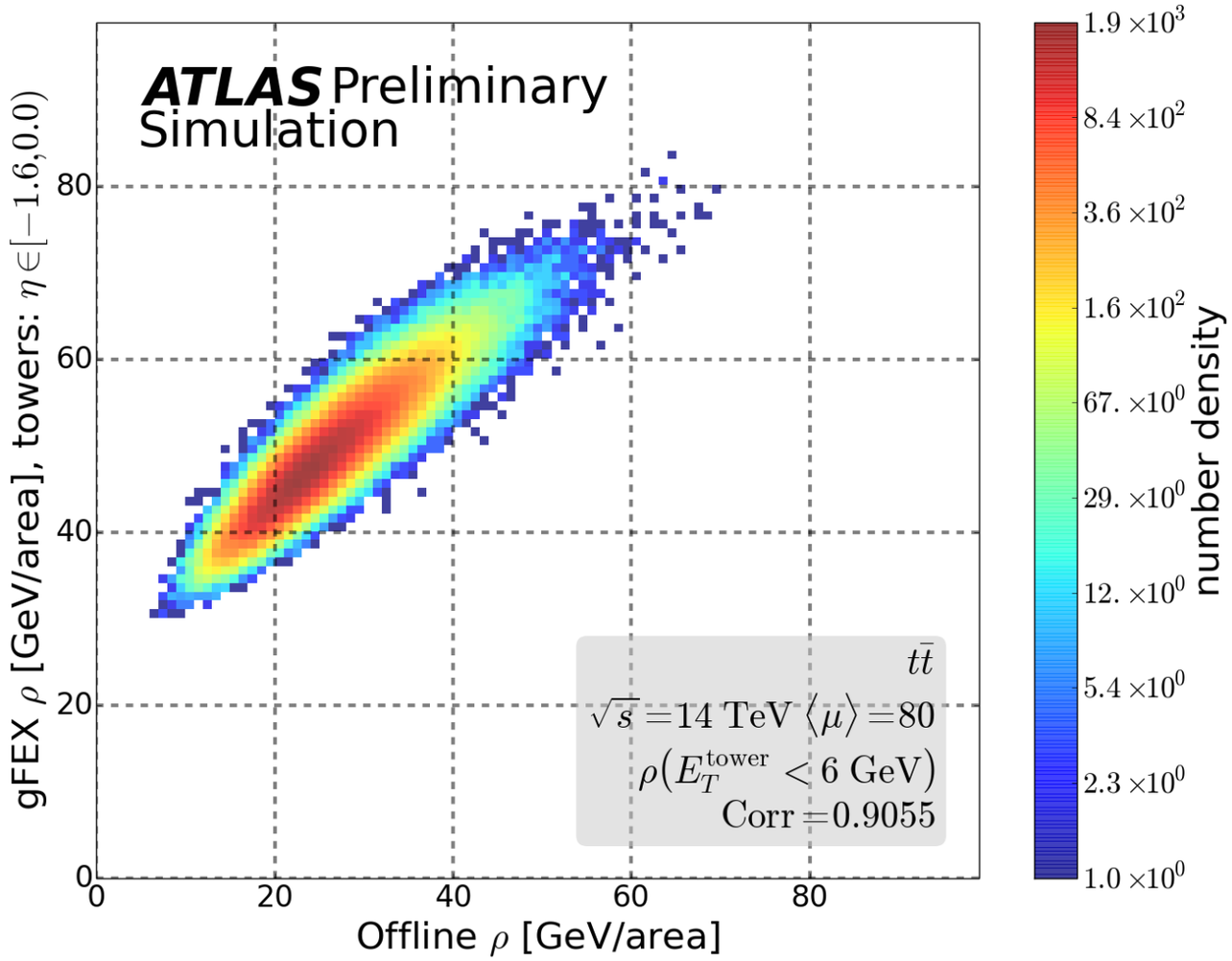
What's inside?

Algorithms Run On FPGAs (x3)

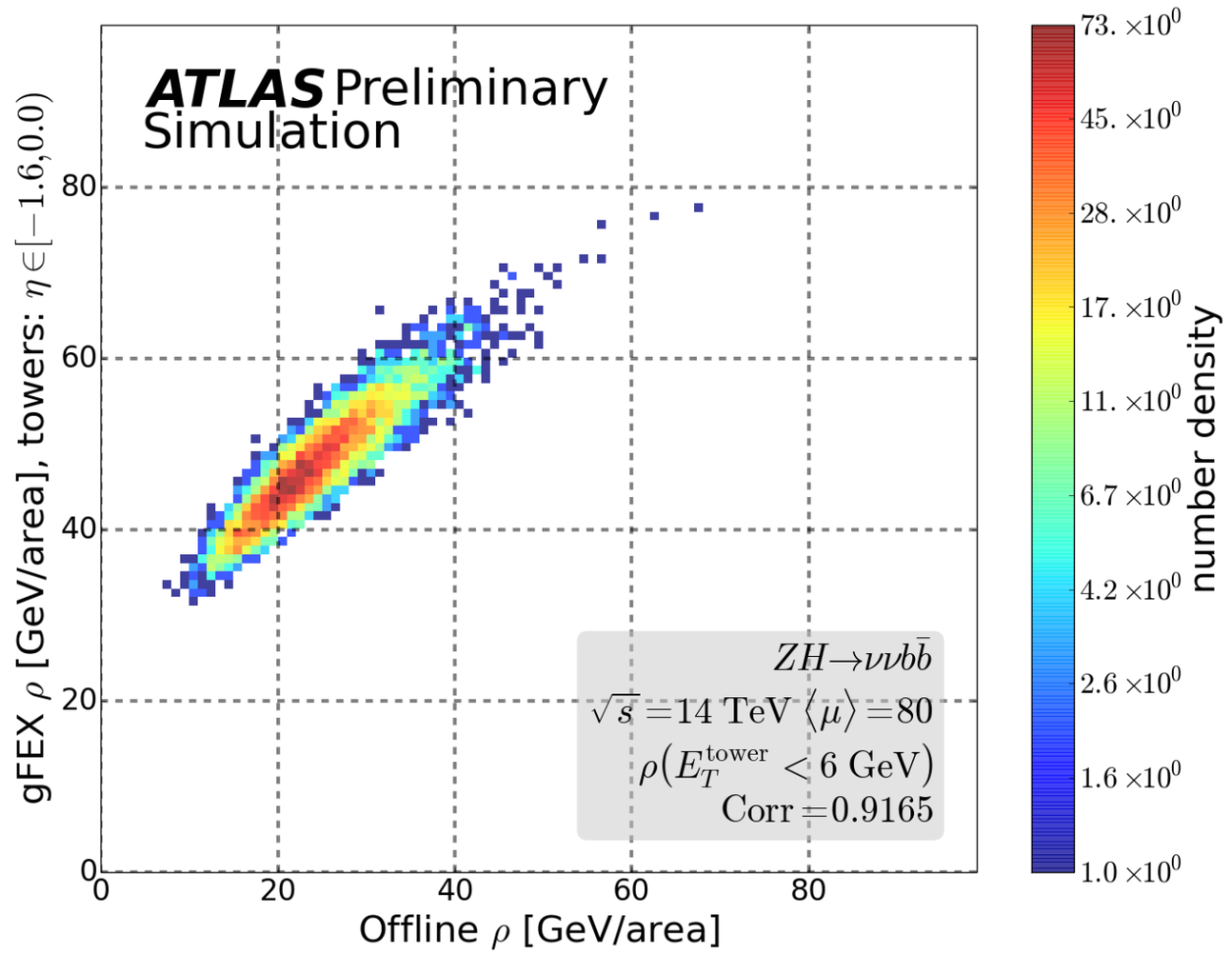


Pile-up Energy Density (ρ) Calculations in the gFEX at the Level 1 Trigger

$t\bar{t}$



$ZH \to \nu\nu b\bar{b}$

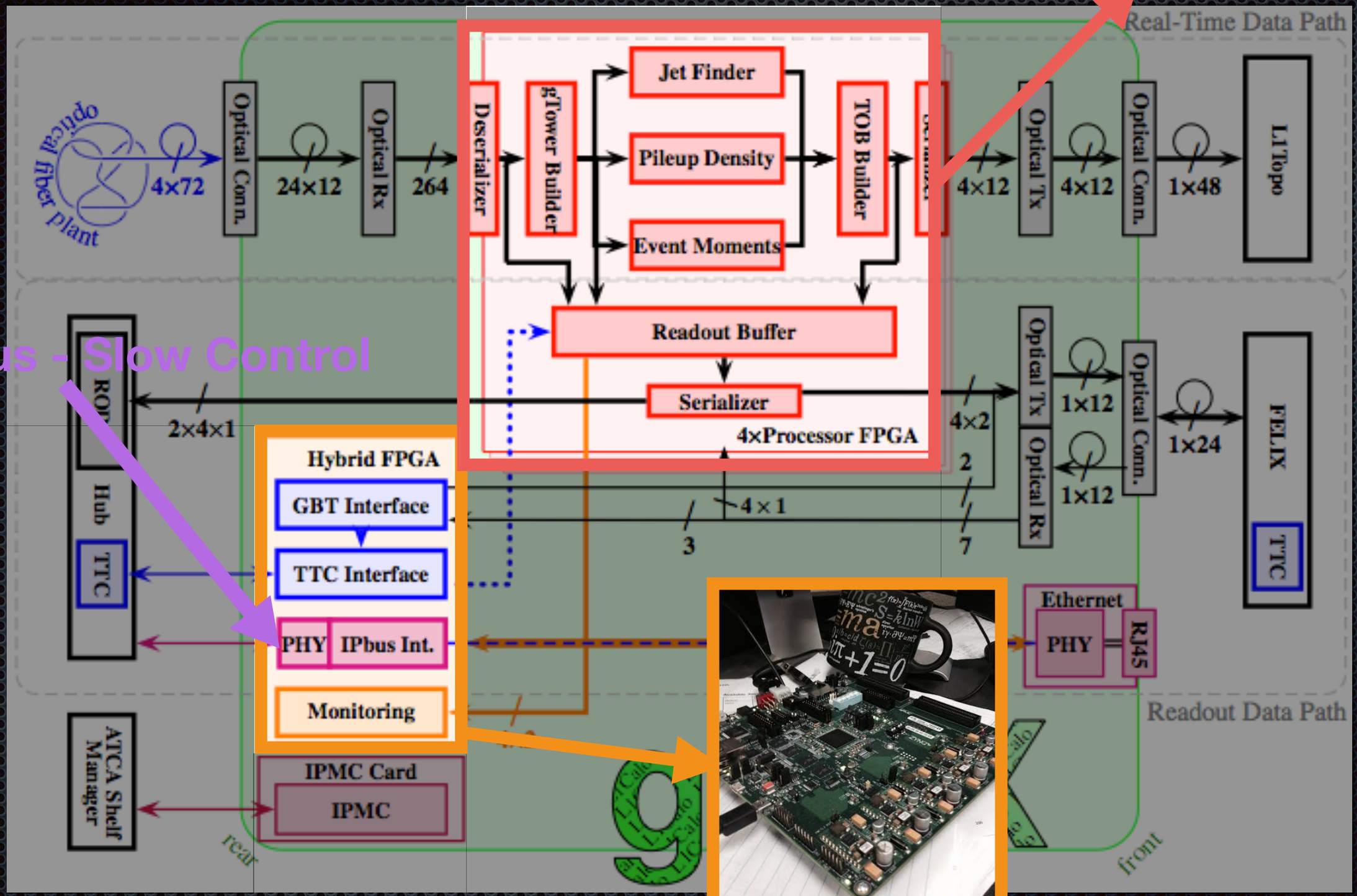


How does our simplified calculation of ρ match up to the corresponding offline calculation?

- Correlation between offline ρ and simplified online ρ using gFEX
- Online calculation independent of physics processes we're studying (it shouldn't and it doesn't).

What's inside?

Algorithms Run On FPGAs (x3)

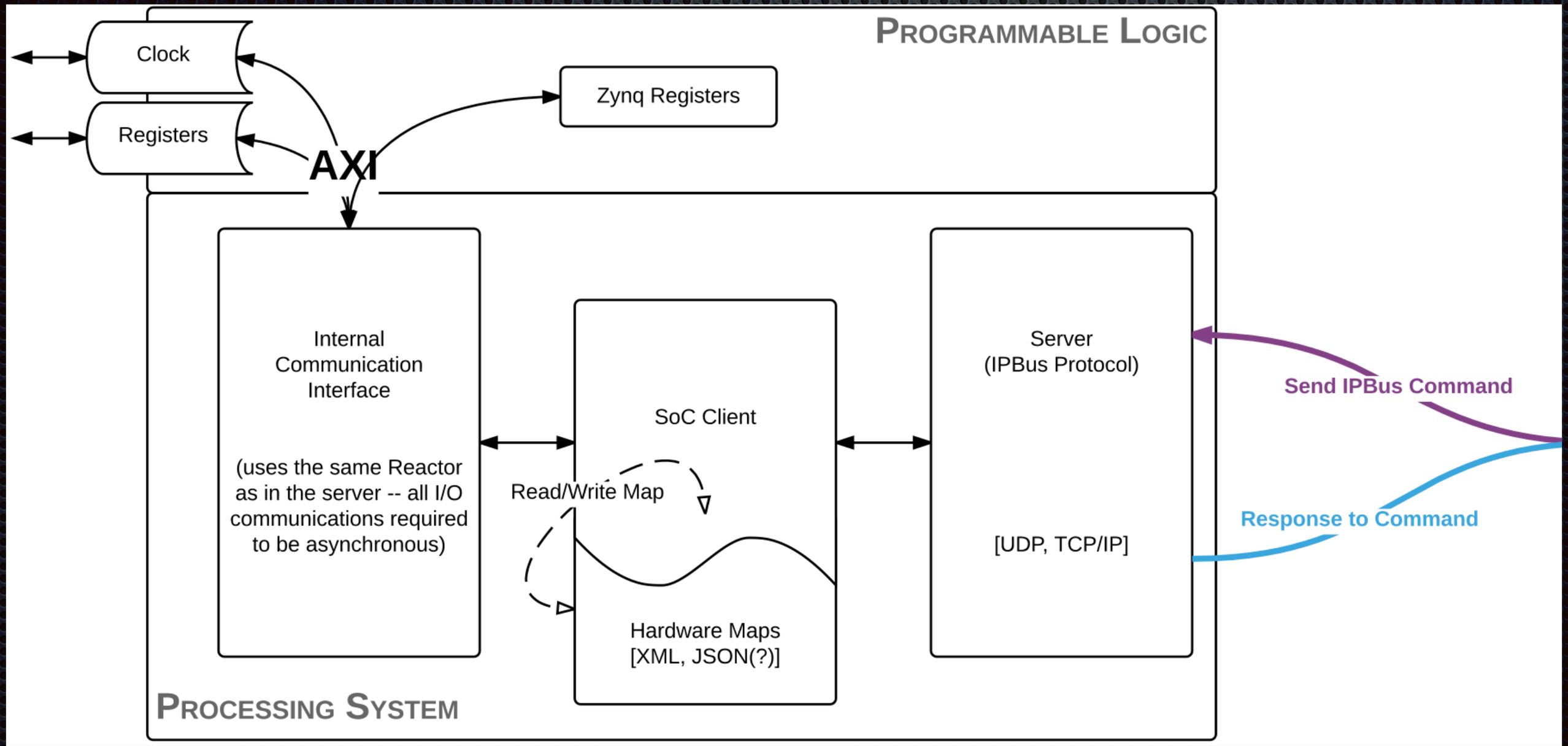


UChicago: FPGA + Embedded Processor— Slow Control and Monitoring

Ironman - Software Package

- The slow control and monitoring infrastructure for the Embedded Processor will be in a python package called **ironman**.
  - https://github.com/kratsg/ironman
  - Documentation: https://iron-man.readthedocs.org/
- The goal:
  - Make it as easy as possible for someone to put their pieces in to the general framework while maintaing the overall procedure
    - custom communication protocols for reading/writing various hardware components
    - custom hardware maps specifying the layout of the entire board

Software Logic Overview



- Packet objects are passed between blocks of logic
- Parsing/building packets, reading hardware maps, managing the server (UDP, TCP, serial, etc…), handling event triggers — **all are done for you**

Server

- The simplest portion of the framework
  - event-driven
  - starts the callback chain
- UDP and TCP are supported

```
from ironman.server import ServerFactory
from ironman.packet import IPBusPacket
from twisted.internet import reactor
from twisted.internet.defer import Deferred

reactor.listenUDP(8888, ServerFactory('UDP',
    lambda: Deferred().addCallback(IPBusPacket)
                      .addCallback(j)
                      .addCallback(buildResponsePacket)
    )
)
reactor.listenTCP(8888, ServerFactory('TCP',
    lambda: Deferred().addCallback(IPBusPacket)
                      .addCallback(j)
                      .addCallback(buildResponsePacket)
    )
)

reactor.run()
```

Hardware Definitions

```
manager = HardwareManager()
manager.add(HardwareMap(file('xadc.xml').read(), 'xadc'))
```

- Given XML files defining the hardware for the board with address maps…
  - centralized all addresses into a single “map” (manager) that knows which map owns a specific address
  - like GPS navigation - ask for an address and it tells you which route you need to take

Jarvis

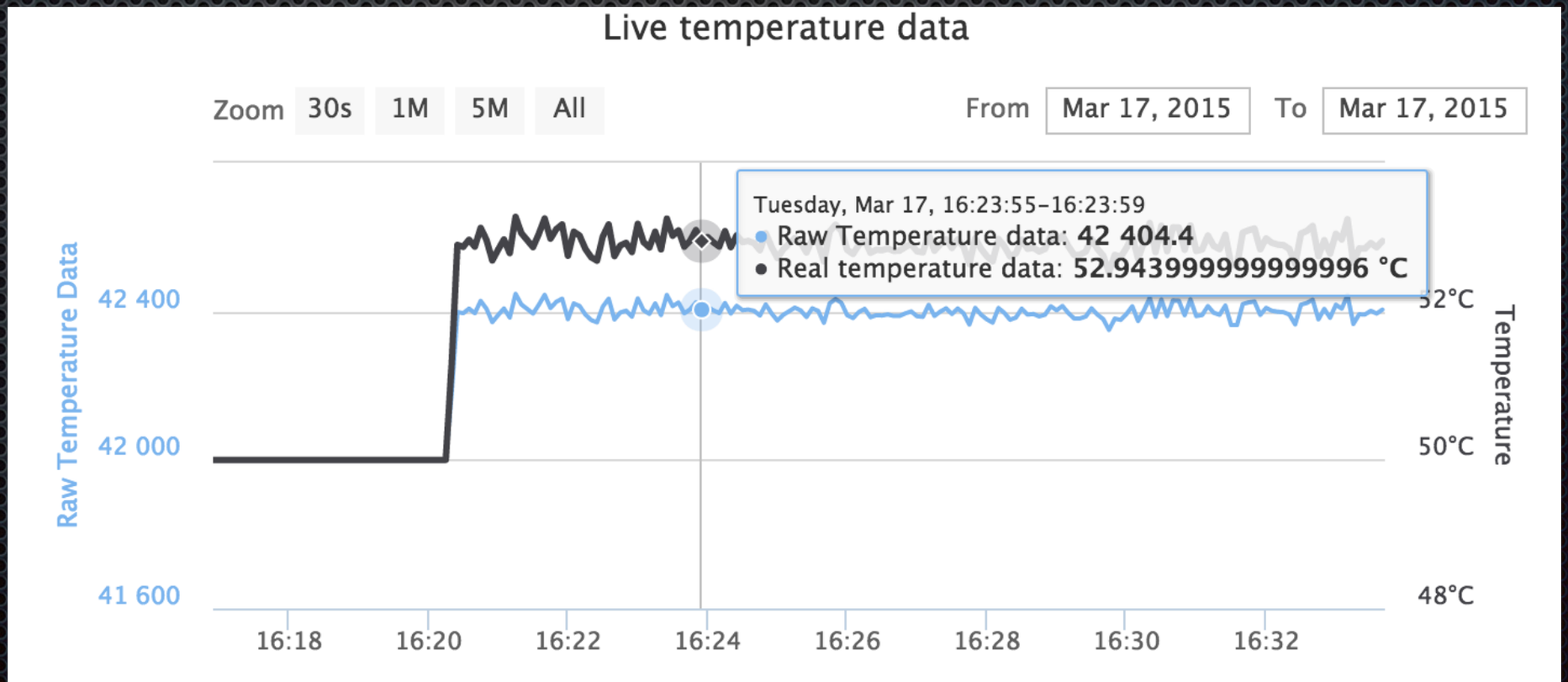
```
from ironman.communicator import Jarvis, ComplexIO
j = Jarvis()
j.set_hardware_manager(manager)

@j.register('xadc')
class XADCController(ComplexIO):
    __base__ = "/sys/devices/soc0/amba@0/f8007100.ps7-xadc/iio:device0/"
    __f__ = {
                0:   __base__+"in_temp0_offset",
```

- Think of Jarvis as your really scrappy OS
  - Manages all drivers (controllers) by associating them with routes
  - Each driver is responsible for implementing reads and writes for a given address

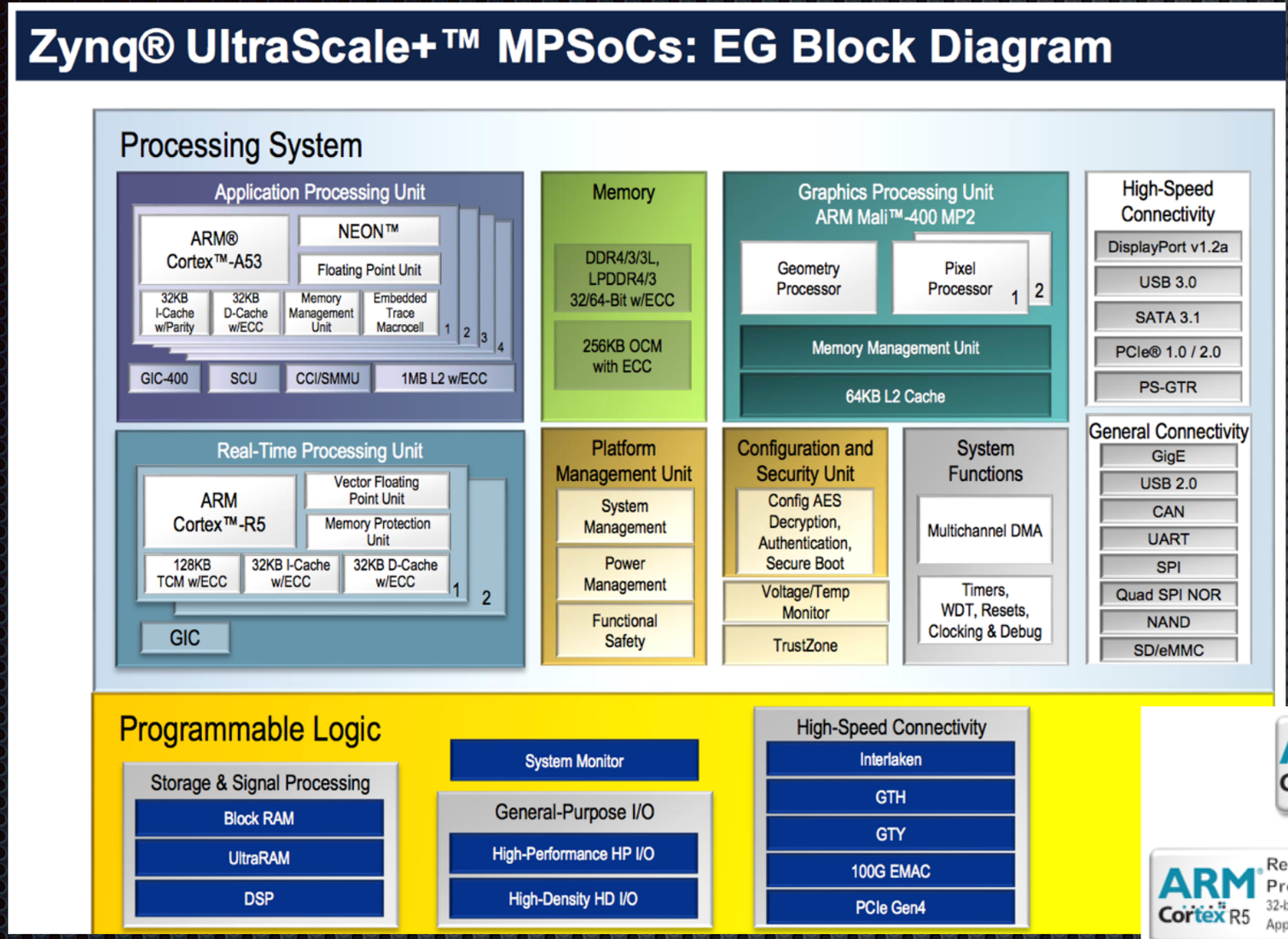
Temperature Monitoring

Time	From IP	From Port	To IP	To Port	Method	Error	ASCII
2:14:24.971 pm	192.168.0.15	8888	You	55056	UDP		\00\00\f0\00\00\01\0f123.
2:14:24.964 pm	You	55056	192.168.0.15	8888	UDP		\f0\00\00 \0f\01\00\00\02\00\00\00
2:14:06.498 pm	192.168.0.15	8888	You	55056	UDP		\00\00\f0\00\00\02\0f123.0407
2:14:06.490 pm	You	55056	192.168.0.15	8888	UDP		\f0\00\00 \0f\02\00\00\02\00\00\00
2:14:02.385 pm	192.168.0.15	8888	You	55056	UDP		\00\00\f0\00\00\02\0f2443\n
2:14:02.378 pm	You	55056	192.168.0.15	8888	UDP		\f0\00\00 \0f\02\00\00\01\00\00\00
2:13:55.705 pm	192.168.0.15	8888	You	55056	UDP		\00\00\f0\00\00\02\0f-2219\n
2:13:55.698 pm	You	55056	192.168.0.15	8888	UDP		\f0\00\00 \0f\02\00\00\00\00\00\00

```
Writing ' \x00\x00\xf0\x00\x00\x02\x0f2440\n'
Writing ' \x00\x00\xf0\x00\x00\x02\x0f-2219\n'
Writing ' \x00\x00\xf0\x00\x00\x02\x0f2443\n'
Writing ' \x00\x00\xf0\x00\x00\x02\x0f123.0407'
Writing ' \x00\x00\xf0\x00\x00\x01\x0f123.'
```

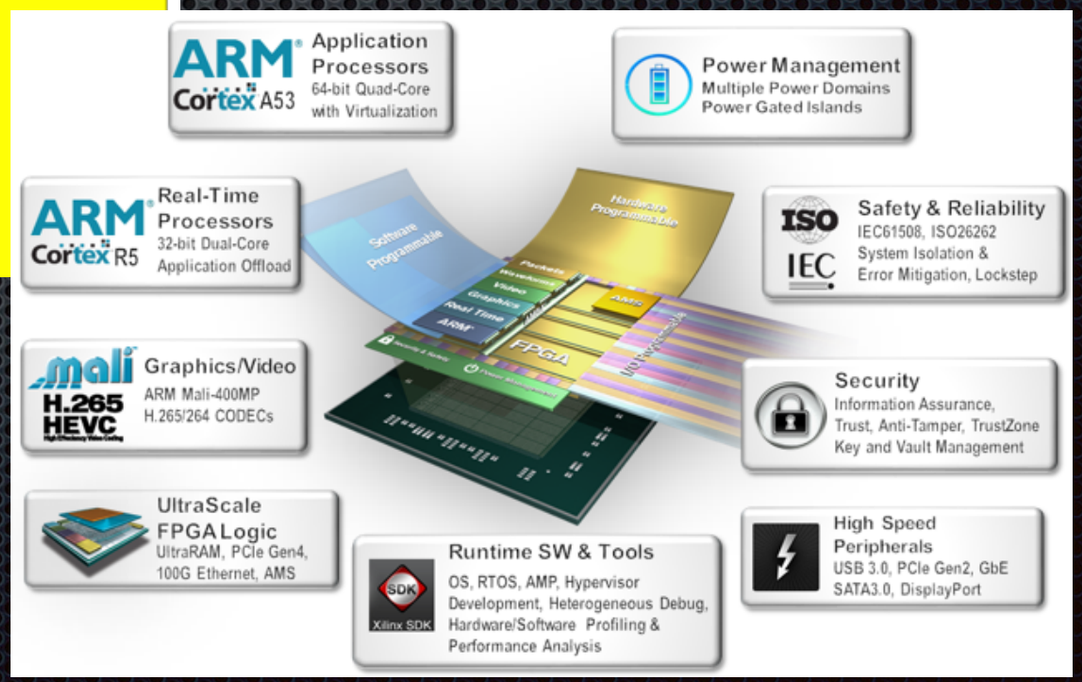


Using bleeding-edge technology



The Zynq UltraScale+ MPSoC

- 4-core ARM Cortex-A53 APU
- 2-core Cortex-R5 RPU
- ARM Mali-400 MP2 GPU



Want gFEX to be useful through to the next generation of physicists (~20 years)

Timeline

- Hardware and Firmware Review: Finished
- Completed design of gFEX v3 (PCB arrived recently)
  - v1: 1 Virtex 7 pFPGA
  - v2: two boards, one w/ 1 pFPGA and one w/ 3 pFPGas
- Production Readiness Review in Fall 2017
- Commission v3 gFEX Module starting early 2018

The Takeaway

- **What are we doing?**
  - Building a large-R jet trigger for Run 3 and beyond
- **Why are we doing it?**
  - Maintain trigger efficiency on interesting events as LHC moves towards higher and higher luminosity
- **What makes gFEX special?**
  - Full calorimeter on a single board with an embedded operating system for slow control and monitoring

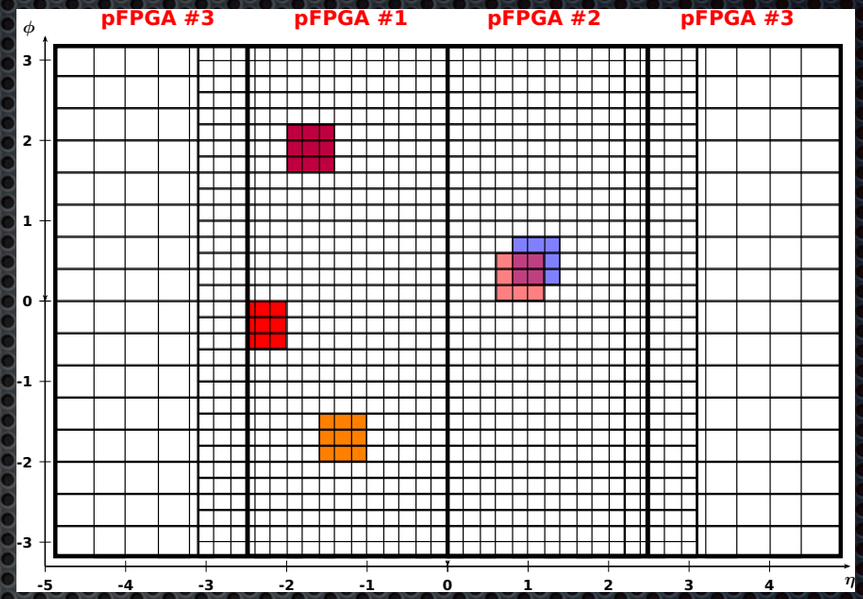
Acronyms

- CTP — Central Trigger Processor
- gFEX / jFEX / eFEX — (global / jet / electron), Feature Extraction Module
- HW / SW / FW — Hardware / Software / Firmware
- L1 — Level 1
- L1Calo / L1Topo — Level-1, (Calorimeter / Topological Processor)
- LAr — Liquid Argon Calorimeter
- LHC — Large Hadron Collider
- TileCal — Tile Calorimeter

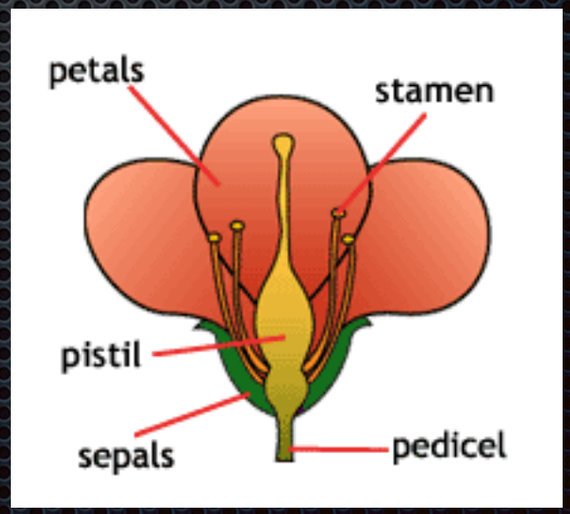
Backup

Object Definitions

- **Small-R Jets**
  - **features**: Anti-Kt R=0.4 built from calo clusters
  - **location**: |eta| < 2.5 (post-selection)
- **gTowers**
  - **features**: built from super cells formed from ***fully simulated L1Calo trigger information***
  - **location**: central, |eta| < 2.4 **area**: 0.2 (eta) x 0.2 (phi)
- **gBlocks**
  - **features**: built from gTowers (one pistil at center and up to 8 stamens)
  - **location**: central, built from central gTowers *(excluding those with 0.1x0.2 in area)*
  - **area**: multiple values ≤ 0.36, depends on where the pistil is located; nominally 0.6 (eta) x 0.6 (phi)



gBlock Cartoon



The HW Map

```
41 lines (40 sloc)  |  2.15 KB
```

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<node id="TOP">
    <node id="temperature" address="0x00000000"
        <node id="offset" address="0x0" descript
        <node id="raw" address="0x1" descriptior
        <node id="scale" address="0x2" descripti
    </node>
    <node id="vccint" address="0x00000010" descr
        <node id="raw" address="0x1" descriptior
        <node id="scale" address="0x2" descripti
    </node>
```

```
@j.register('xadc')
class XADCController(ComplexIO):
    __base__ = "/sys/devices/soc0/amba@0/f8007100.ps7-xadc/iio
    __f__ = {
                0:   __base__+"in_temp0_offset",
                1:   __base__+"in_temp0_raw",
                2:   __base__+"in_temp0_scale",
                17:  __base__+"in_voltage0_vccint_raw",
                18:  __base__+"in_voltage0_vccint_scale",
                33:  __base__+"in_voltage1_vccaux_raw",
                34:  __base__+"in_voltage1_vccaux_scale",
                49:  __base__+"in_voltage2_vccbram_raw",
                50:  __base__+"in_voltage2_vccbram_scale",
                65:  __base__+"in_voltage3_vccpint_raw",
                66:  __base__+"in_voltage3_vccpint_scale",
                81:  __base__+"in_voltage4_vccpaux_raw",
                82:  __base__+"in_voltage4_vccpaux_scale",
                97:  __base__+"in_voltage5_vccoddr_raw",
                98:  __base__+"in_voltage5_vccoddr_scale",
                113: __base__+"in_voltage6_vrefp_raw",
                114: __base__+"in_voltage6_vrefp_scale",
                129: __base__+"in_voltage7_vrefn_raw",
                130: __base__+"in_voltage7_vrefn_scale"
    }
```

https://github.com/kratsg/ironman/blob/master/workbench/xadc.xml